Portland State University
PDXScholar

Dissertations and Theses

Dissertations and Theses

2-4-1997

A Simplified Approach to Reduce Blocking and Ringing Artifacts in Transform-Coded Images

Jianping Hu
Portland State University

Follow this and additional works at: https://pdxscholar.library.pdx.edu/open_access_etds

 Part of the Computer Engineering Commons, and the Other Electrical and Computer Engineering Commons

Let us know how access to this document benefits you.

Recommended Citation
Hu, Jianping, "A Simplified Approach to Reduce Blocking and Ringing Artifacts in Transform-Coded Images" (1997). *Dissertations and Theses.* Paper 5383.

This Thesis is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. For more information, please contact pdxscholar@pdx.edu.

THESIS APPROVAL

The abstract and thesis of Jianping Hu for the Master of Science in Electrical and Computer Engineering were presented November 19, 1996, and accepted by the thesis committee and the department.

COMMITTEE APPROVALS: [redacted]

Fu Li, Chair

[redacted]

Andrew Fraser

[redacted]

Kwok-Wai Tam
Representative of the Office of Graduate Studies

DEPARTMENT APPROVAL: [redacted]

Rolf Schaumann, Chair
Electrical and Computer Engineering

****

ACCEPTED FOR PORTLAND STATE UNIVERSITY BY THE LIBRARY

by [redacted] on 4 February 1997

ABSTRACT

An abstract of the thesis of Jianping Hu for the Master of Science in Electrical and Computer Engineering presented November 19, 1996.

Title: A SIMPLIFIED APPROACH TO REDUCE BLOCKING AND RINGING ARTIFACTS IN TRANSFORM-CODED IMAGES

Presently Block-based Discrete Cosine Transform (BDCT) image coding techniques are widely used in image and video compression applications such as JPEG and MPEG. At a moderate bit rate, BDCT is usually a quite satisfactory solution to most of practical coding applications. However, for high rate compression it produces noticeable blocking and ringing artifacts in the decompressed image. It has been an active research area for a decade for reducing these artifacts.

In this thesis, a novel post-processing algorithm is proposed to remove the blocking and ringing artifacts at low bit rate. It is non-iterative and uses both spatial and transform domain approaches. The main steps in this algorithm include block classification, boundary low-pass filtering and mid-point interpolation, edge detection and region merge, and DCT coefficient constraint. The improvement is demonstrated both subjectively and objectively. The extension from still images to video is also discussed at the end.

A SIMPLIFIED APPROACH TO REDUCE BLOCKING AND RINGING ARTIFACTS IN TRANSFORM-CODED IMAGES

by

JIANPING HU

A thesis submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
ELECTRICAL AND COMPUTER ENGINEERING

Portland State University
1997

Acknowledgements

I wish to express my appreciation to my advisor and the chair of my thesis committee, Dr. Fu Li, who provided many suggestions to this research as well as guidance and support. I would also like to thank the other members of my committee, Dr. Andrew Fraser and Dr. Kwok-Wai Tam.

I want to express my gratitude to the staff of the Electrical Engineering Department at Portland State University for their help and support all the time.

My appreciation is extended to all my co-researchers at Image Processing Group and all my friends at Portland State University for their help throughout the completion of this thesis.

Finally, my special appreciation to my family for their support during all stages of my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the recent years, digital image and video has become a priority choice for many applications, such as digital camera, photo CD, facsimile and computer multimedia. The critical problem has to be solved is the digital image and video compression, because the digital representation of image and video requires large number of bits. For example, a typical low-resolution, TV quality, color video image (512×512 pixels/color, 8 bits/pixel, and 3 colors) requires 6×10^6 bits. A 24×36-mm (35-mm) negative photograph scanned at $12\mu m$ (3000×2000 pixels/color, 8 bits/pixel, and 3 colors) requires 144×10^6 bits. Apparently, the storage and the transmission of even a few images without compression could be a problem.

The good thing is that image data are often highly redundant and/or irrelevant. Redundancy refers to the statistical properties of images, and irrelevancy relates to an observer viewing of an image. There are three types of redundancy in digital images: spatial correlation between neighboring pixel values, spectral correlation between color planes or spectral bands, and temporal correlation between neighboring frames in a sequence of images. Irrelevancy connects with the property of human visual system (HVS). HVS exhibits sensitivity variations as a function of spatial frequency, orientation, light level, surrounding signals, etc. Scene content

and noise, image size and viewing distance, display characteristics, and the observer will affect irrelevancy.

Image and video compression can be achieved by removing these redundancy and irrelevancy. There are many approaches to image compression, but they can be categorized into two fundamental groups: *lossless* and *lossy*. Lossless compression, also called as *reversible* or *bit-preserving* compression, can preserve all the information in the image. The decompressed image is exactly the same as the original image. A modest amount of compression ratio, generally in the range of 1.5:1 to 4:1, can be achieved because only the statistical redundancy is exploited during compression. Thus, lossless compression limits its application in the areas such as medical image, where the quality of the image is the most important thing to be considered. On the other hand, lossy compression is widely accepted by many other applications, such as consumer products. In these applications it is desirable to compress the image as much as possible. It is only necessary that the quality of the image is good enough for visual or machine analysis, and loss of some information about the image is acceptable.

For the lossless compression, Differential Pulse Code Modulation (DPCM) is a popular choice. This method is based on predicting the pixel being considered from its adjacent pixels to get the differential image. The differential image typically has a largely reduced variance compared to the original image and is significantly less correlated. The differential image is usually entropy encoded to achieve lossless compression. One approach is variable-length coding, such as Huffman coding. The other choices for the lossless compression are bit plane encoding and lossy plus

lossless residual encoding. But they have little chance to be used in the practical applications.

There are even more methods for the lossy image compression. Lossy predictive coding and transform coding has been used in many applications. Subband coding, Wavelet coding and fractal coding are still in the research stage. Among various lossy compression methods, transform-based compression is by far the most popular choice both in still image compression and video compression. Some well-known transforms are Karhunen-Loève Transform, Discrete Fourier Transform, Discrete Cosine Transform, Walsh-Hadamard Transform and Haar Transform. But only Block-based Discrete Cosine Transform (BDCT) is the dominant one because of its near-optimum energy compaction property and availability of fast algorithms and hardware. Most of the current standards such as JPEG, MPEG and H.261 use BDCT.

In the BDCT coding, quantization in some form is used to discard unimportant information and to reduce the amount of data to be transmitted or stored. The BDCT based coding can successfully compress images by a factor around 10 with nearly no perceptible effects. However, some well-known artifacts arise at low bit rate compression. The two most obvious artifacts in a low bit coded image are "blocking" and "ringing" effects. Dividing the image into blocks prior to coding causes blocking effect—discontinuities between adjacent blocks. And discarding high frequency DCT coefficients during quantization causes ringing effect—contouring along sharp edges.

In the past two decades, a variety of efforts have been made to remedy these

problems, primarily in two major categories:

1. At the encoding end, different encoding schemes have been proposed to avoid such artifacts. In [1], a block overlap method was proposed to reduce blocking effect. In [2], the edge blocks are detected from non-edge blocks and then these two types of blocks are coded differently to remove ringing effect. A DC calibration scheme appeared in [3] uses the anchor blocks and code their DC components error-free for blocking effect removal.

2. At the decoding end, different post-processing algorithms have been suggested to reduce such artifacts. In [1], a simple low-pass filter was used to smooth the unwanted discontinuities at or near the block boundaries. Edge based adaptive filtering post-processor appeared in [4] and [5]. Another approach for reducing coding artifacts is to use image restoration theory. Proposed methods include convex projections(CP) [6, 7, 8], and maximum *a posteriori* (MAP) estimation [9].

Despite various progress reported at the encoding end, changing encoding schemes means to abandon well-accepted JPEG or MPEG standards, which makes research on these progress strictly in academia. In other words, practical implementation and commercial adoption are still beyond the horizon.

In contrast, post-processing approaches at the decoding end have good potential to be integrated into image and video communications, as they are applied to JPEG and MPEG standards.

In this thesis, a new post-processing artifact removal algorithm which lies at

the decoding end is proposed. The new approach uses both spatial and transform domain methods to remove blocking and ringing artifacts at low bit rate. And this algorithm can achieve the improvement under both objective and subjective criteria.

The rest of this thesis is organized as follows:

In Chapter 2, general image compression methods are discussed. The topics include Transform, Quantization and Coding. This chapter serves as the basement of the following chapters in this thesis.

In Chapter 3, Block-based Discrete Cosine Transform and the JPEG Baseline algorithm are explained in more detail.

In Chapter 4, JPEG lossy image compression artifacts are showed and some existing approaches to reduce the compression artifacts are discussed briefly.

In Chapter 5, the proposed algorithm is explained step by step. The main steps include block classification, boundary low-pass filtering and mid-point interpolation, edge detection and region merge, and DCT coefficients constraint.

In Chapter 6, the computer simulation results of the algorithm is given. The improvement is demonstrated both subjectively and objectively.

In Chapter 7, the conclusion of this thesis is given.

Chapter 2

An Overview of Transform-based Image Compression

2.1 Introduction

In this chapter the transform-based image compression steps are explained in general. This provides the background for the following chapters about JPEG standard and artifact removal algorithm.

2.2 Transform-based Image Compression

A typical lossy image compression technique includes these steps: Transformation or Decomposition, Quantization, Symbol Encoding. The block diagram is shown in Fig. 2.1. In this diagram, the *Transformation or Decomposition* box can use different kinds of existing techiques, e.g., DPCM prediction; DCT; Subband decomposition. The *Quantization* box can choose from uniform quantizer, Lloyd-Max quantizer, or Entropy-constrained quantizer, etc. Huffman coding and Arithmetric coding are the two most popular choices for the *Symbol Encoding* step.

In the following, each of these steps will be described and the attention will go

to the transform-based image compression.

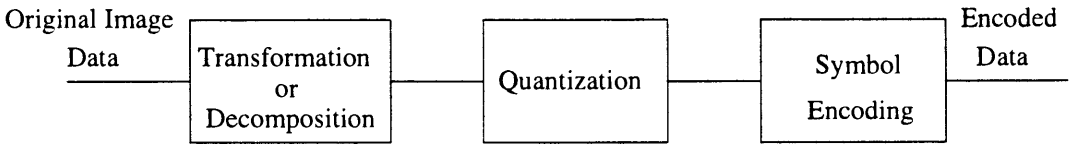


Figure 2.1: Generalized Lossy Image Coding Scheme

2.2.1 Image Transforms

Image transforms usually refers to a class of unitary matrices used for representing images. Just as a one-dimensional signal can be represented by an orthogonal series of *basis functions*, an image can also be expanded in terms of a discrete set of basis arrays called *basis images*. These basis images can be generated by unitary matrices. Most unitary transforms have a tendency to pack a large fraction of the average energy of the image into a relatively few components of the transform coefficients. Since the total energy is preserved, this means many of the transform coefficients will contain very little energy. This is a very useful property for the image compression. The reason will be explained in the next two sections.

The number of multiplications and additions required to compute the transform coefficients for an $N \times N$ image is $O(N^4)$ and can be reduced to $O(N^3)$ if the two-dimensional transform is separable. Thus for the real world application an image is usually divided into small size blocks, usually square, before transformation. Then each block is taken transformation independently.

Several orthogonal transforms encountered in image processing are: Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), Discrete Sine Transform, the Hadamard Transform, the Haar Transform, the Slant Transform, and the Karhunen-Loève Transform. The choice of the transform is based on its energy compaction and decorrelation ability, its performance in image compression based on the mean square criterion and the existence of fast algorithms. Considering the above criterias and the existing technology, DCT surpassed the other transforms and became the dominant one in the area of image and video compression.

For image processing applications, the forward 2-D DCT of an $n \times n$ block of pixels is often defined as

$$F(u,v) = \frac{4C(u)C(v)}{n^2} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} f(j,k) \cos\left[\frac{(2j+1)u\pi}{2n}\right] \cos\left[\frac{(2k+1)v\pi}{2n}\right] \quad (2.1)$$

and the inverse 2-D DCT is defined as

$$f(j,k) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} C(u)C(v)F(u,v) \cos\left[\frac{(2j+1)u\pi}{2n}\right] \cos\left[\frac{(2k+1)v\pi}{2n}\right] \quad (2.2)$$

where

$$C(w) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } w = 0 \\ 1 & \text{for } w = 1, 2, \ldots, n-1 \end{cases} \quad (2.3)$$

The Discrete Cosine Transform causes the most of the energy of the input matrix to be concentrated in the upper left corner of the transformed matrix, as shown in Fig. 2.2.

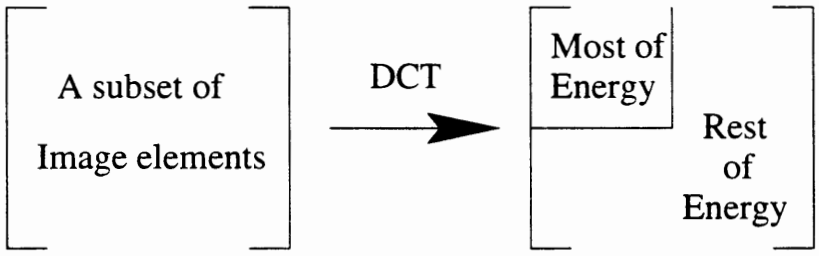


Figure 2.2: The effect of DCT on the input matrix

2.2.2 Quantization

A quantizer is essentially a staircase function that maps the possible input values into a smaller number of output levels. For the purposes of image compression, quantization reduces the number of symbols that need to be encoded at the expense of introducing errors in the reconstructed image. The quantization acts as a control knob that trades off image quality for bit rate.

Different quantizers can be classified into two categories: *scalar quantization* (SQ) and *vector quantization* (VQ). Scaler quantization is that each signal value is individually quantized. And vector quantization is that a block of signal values is jointly quantized. For the scaler quantization, uniform quantizer, Lloyd-Max or MMSE quantizer, and entropy-constrained quantizer are the three quantizers used more often than the others at present. So let us briefly discuss the scalar quantizer.

A scalar quantizer is actually a staircase function that maps many input values (the input values can be continuous) into a smaller, finite number of discrete output levels. Suppose a scalar quantizer has N levels, the input signal x has random value and is mapped into a discrete variable x' that belongs to a finite set $\{r_i, i =$

$0, \ldots, N-1\}$ of real numbers referred to as *reconstruction levels* or quantizer output levels. The range of values x that map to a particular x' are defined by a set of points $\{d_i, i = 0, \ldots, N\}$, referred to as *decision levels*. The quantization rule states that if x lies in the interval $(d_i, d_{i+1}]$, it is mapped or quantized to r_i, which also lies in the same interval. Once quantized, the actual value of the continuous signal can never be reconstructed exactly, which means some information is lost during quantization.

The simplest form of a scalar quantizer is a uniform quantizer where the quantizer decision levels are all of equal length, referred to as the quantizer step size. And the reconstruction levels are the middle point between the two adjacent decision levels, as shown in Fig. 2.3.

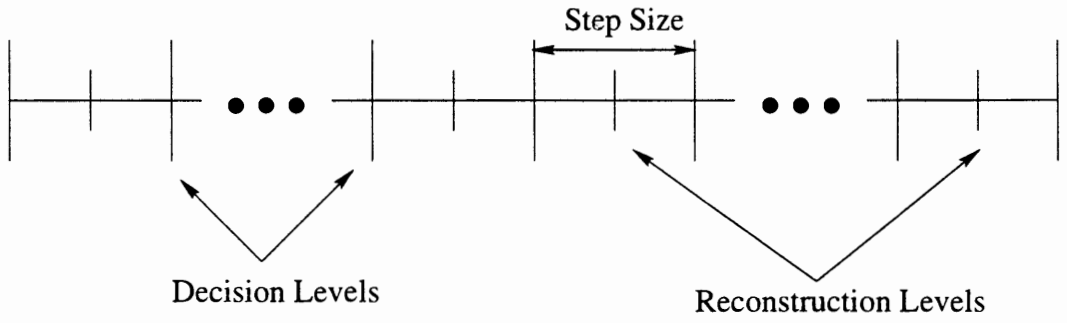


Figure 2.3: The Uniform Quantizer

The difference between the actual input value and the reconstruction value is the quantization error. In some applications, it is desirable to find the quantizer that minimizes the quantization distortion for a given number of quantizer output

levels. For a minimum-mean-square error (MMSE) distortion criterion, this quantizer is known as the Lloyd-Max quantizer. Consider a signal x with a probability distribution function (pdf) $p(x)$, for a given number of quantizer output levels N, the optimum MMSE quantizer has reconstruction levels r_i, and decision levels d_i that minimize

$$\epsilon = \sum_{i=1}^{N} \int_{d_i}^{d_{i+1}} (x - r_i)^2 p(x) dx \tag{2.4}$$

The solution to the above minimization problem is

$$d_j = \frac{r_{j-1} + r_j}{2} \tag{2.5}$$

$$r_j = \frac{\int_{d_i}^{d_{i+1}} x p(x) dx}{\int_{d_i}^{d_{i+1}} p(x) dx} \tag{2.6}$$

Each decision level is half-way between the two neighboring reconstruction levels, and each reconstruction level is at the centroid of the *pdf* that is enclosed by the decision region, as shown in Fig. 2.4.

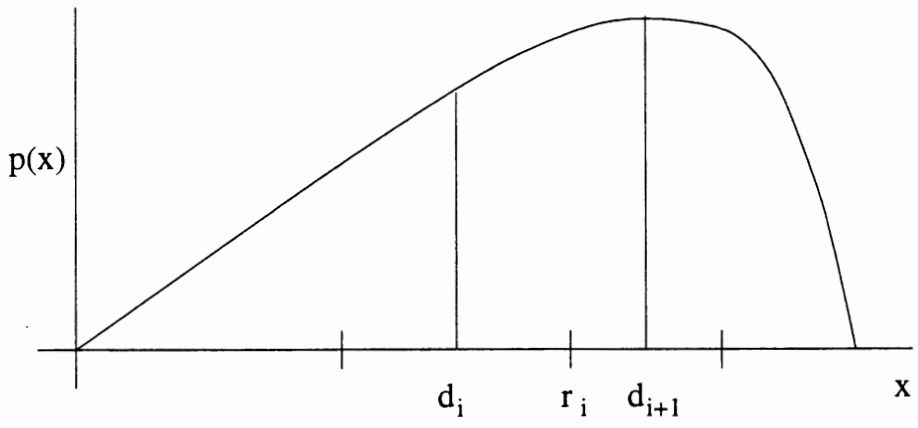


Figure 2.4: The Lloyd-Max Quantizer

The Lloyd-Max quantizer aims at minimizing the MSE for a given number of output quantization levels, and is useful when the output levels are coded with fixed-length codes. When variable-length coding is permitted, the final bit rate is determined by the entropy of the quantizer levels rather than by the number of levels. For example, a uniform quantizer may have a lower entropy, despite a larger number of output levels, compared to a Lloyd-Max quantizer operating with the same distortion. This leads to another criterion for quantizer design known as *entropy-constrained quantization*(ECQ), where the quantization error is minimized subject to the constraint that the entropy of the quantizer output levels has a prescribed value.

It has been theoretically established that for memoryless sources and at high output bit rates, the optimal entropy-constrained SQ quantizer is a uniform quantizer. It has been experimentally shown that even at low bit rates, for the MSE distortion and for a variety of memoryless sources, the performance of the uniform quantizer is virtually indistinguishable from that of the optimal ECQ. Based on the above results, a uniform scalar quantizer followed by entropy coding should be considered when i) The source is memoryless; ii) The bit rate is moderate to large, and; iii) Variable-length coding of the quantizer levels are permitted.

2.2.3 Symbol Encoding

Definition of Information

Consider a source that emits a sequence of symbols from a fixed finite source alphabet of size n, $S = \{s_1, s_2, \ldots, s_n\}$, where the probability of the occurrence of the symbol s_i is given by $p(s_i)$. Furthermore, assume that successive symbols emitted from the source are statistically independent.

If symbol s_i occurs, an amount of information equal to

$$I(s_i) = \log_2 \left(\frac{1}{p(s_i)} \right) \text{ bits} \tag{2.7}$$

is provided.

The probability of occurrence of a source symbol is $p(s_i)$, so the average amount of information obtained per source symbol from the source is

$$H(S) = \sum_{i=1}^{n} p(s_i) I(s_i) = \sum_{i=1}^{n} p(s_i) \log \frac{1}{p(s_i)} \tag{2.8}$$

This is called the *entropy* of the source.

Let us take a look an example: a source with an alphabet consisting of four letters $S = \{s_1, s_2, s_3, s_4\}$, where $p(s_1) = 0.60$, $p(s_2) = 0.30$, $p(s_3) = 0.05$, $p(s_4) = 0.05$. The entropy of this source is

$$H(S) = 1.40 \text{bits/symbol} \tag{2.9}$$

Obviously, the entropy of a source attains its maximum value when all the source symbols are equally probable and that value is $\log_2 n$ bits, where n is the size of the source alphabet.

The entropy is the average amount of information per source symbol provided by the source. According to Shannon's noiseless source coding theorem, no uniquely decodable code can have an average length less than the entropy by encoding the source with a binary alphabet for transmission or storage.

Variable-Length Coding

From the previous example, we see the entropy of the source S is 1.40. Now we like to use a binary alphabet to efficiently encode this source. The code should have some desired characteristics. For instance, each codeword in the sequence should be instantaneously decodable, i.e., decodable without reference to the succeeding codewords. A necessary and sufficient condition for constructing such codes is that no codeword can be a prefix of some other codeword. Any code satisfying this condition is called a *prefix condition code*. Now we code the source S in two different ways as shown in Table 2.1. The average length of Code I is 2.0 bits/symbol, and the average length of Code II is 1.5 bits/symbol. Apparently, the variable-length code is a more efficient code with an average length is closer to the source entropy.

Huffman Coding

For the previous code example, we may construct other variable-length codes, but none would have an average length shorter than Code II. A code is *compact* (for a given source) if its average length is less than or equal to the average length of all other prefix condition codes for the same source and the same code alphabet. According to this definition, Code II is compact for the source S.

Table 2.1: Fixed-length and variable-length codes for S.

Symbol	Probability	Code I	Code II
s_1	0.60	00	0
s_2	0.30	01	10
s_3	0.05	10	110
s_4	0.05	11	111

A general method for constructing compact codes is proposed by Huffman and is based on the following two principle:

- Consider a source with an alphabet of size α. By combining the two least probable symbols of this source, a new source with $\alpha - 1$ symbols is formed. It can be shown that the codewords for the original source are identical to the reduced source codewords for all symbols that have not been combined. Futhermore, the codewords for the two least probable symbols of the original source are formed by appending (to the right) '0' or '1' to the codeword corresponding to the combined symbol in the reduced source.

- The Huffman code for a source with only two symbols consists of the trival codewords '0' and '1'.

Thus, in order to construct the Huffman code for a given source S, we can repeatedly combine the two least probable symbols until only two symbols are obtained. Then '0' and '1' are assigned to these two symbols. The codewords for the previous stage are obtained by appending a '0' and '1' to the codeword corresponding to the

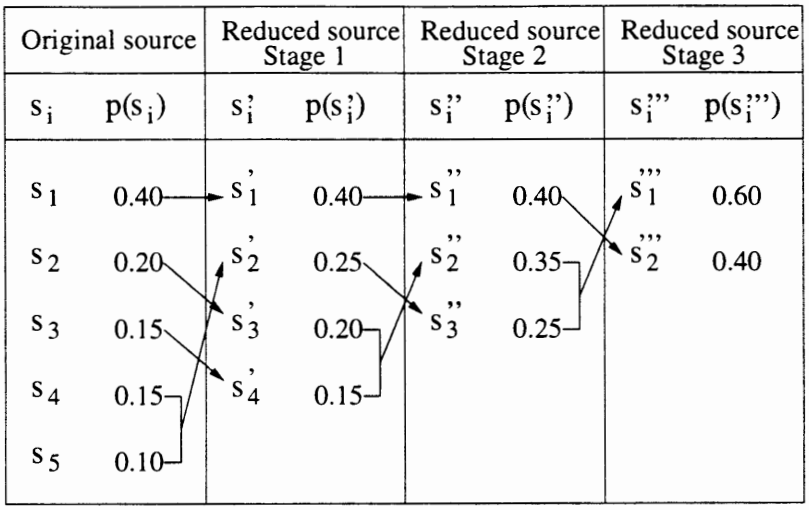
two least probable symbols. This process continues until the Huffman code for the original source is found. Fig. 2.5 shows an example of the reduction and construction process for a source with five symbols.

Modified Huffman Coding

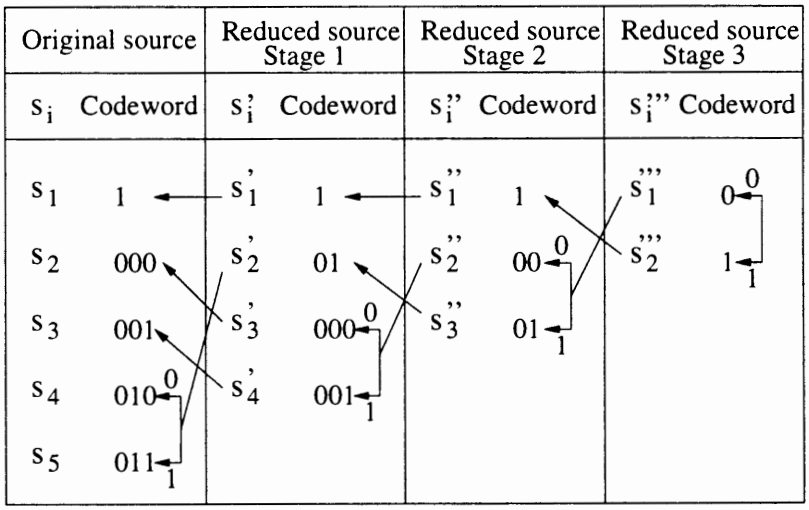
It is not unusual that most of the symbols in a large symbol set have very small probabilities. These symbols will take a large portion of the code book. The length of these symbols is roughly proportional to its information content, $-\log p(s_i)$. It is more convenient to combine the less probable symbols into a symbol called 'ELSE' and design a Huffman code for the reduced symbol set including the ELSE symbol. This procedure is known as the *modified Huffman code.* Whenever a symbol belonging to the ELSE category needs to be encoded, the encoder transmits the codeword for ELSE followed by extra bits needed to identify the actual value within the ELSE category. If the symbols in the ELSE category have a small probability, the loss in coding efficiency (increase in average bit rate) will be very small while the storage requirements and the decoding complexity are substantially reduced.

Run-length Coding

Consider a binary source whose output is coded as the number of 0s between two successive 1s, that is, the length of the runs of 0s are coded. This is called *run-length coding* (RLC). It is useful whenever large runs of 0s are expected. Such a situation occurs in printed documents, graphics, weather maps, and so on. We will see later that the run-length coding is also can been used in the image transform



a) Source reduction process



b) Codeword construction process

Figure 2.5: Huffman code generation

compression where there are large runs of 0s after quantization in the transform domain. In runlength coding, a new message set is constructed based on the runs of 0s and 1s for the binary data. In 1-D runlength coding, the runs of 0s and 1s are variable-length encoded using separate Huffman tables tailored to the statistics of each.

Chapter 3

JPEG Baseline Algorithm For Lossy Image Compression

3.1 Introduction

In this chapter, the steps involved in the JPEG baseline algorithm for lossy image compression are explained. This algorithm is a very successful compression method for continuous-tone still images from the point of view of compression and quality factors.

3.2 JPEG Baseline Algorithm

JPEG (Joint Photographic Experts Group) was formed under the joint auspices of ISO (International Standadization Organization) and ITU-T (International Telecommunications Union) in 1986 to work on an international standard for the compression and decompression of still-frame, continuous-tone, monochrome and color images for many diverse applications. The JPEG standard defines three different coding systems:

1. a lossy *baseline coding system*, which provides a simple and efficient DCT-based algorithm adequate for most image compression applications. It uses Huffman coding, operates only in sequential mode, and is restricted to 8 bits/sample source image precision.

2. an *extended coding system* for greater compression, higher precision, or progressive reconstruction applicatiions. The optional features include 12-bit/sample input, sequential and/or hierarchical progressive build-up, arithmetic coding, adaptive quantization, tiling, still picture interchange file format (SPIFF) and selective refinement.

3. a lossless *independent coding system* for reversible compression, which provides a simple DPCM-based lossless method independent of the DCT to accommodate applications requiring lossless compression.

In this chapter, we will focus on the JPEG baseline coding system. The simplified block diagrams for the encoder and decoder are shown in Fig. 3.1 and Fig. 3.2.

3.2.1 Encoding

Block-based Discrete Cosine Transform

In the baseline system, the original image is first subdivided into pixel blocks of size 8×8, which are processed left to right, top to bottom. In order to facilitate the implementation of the DCT, the values of the 64 pixels in each block or subimage are level shifted by subtracting the quantity 2^{n-1}, where 2^n is the maximum number

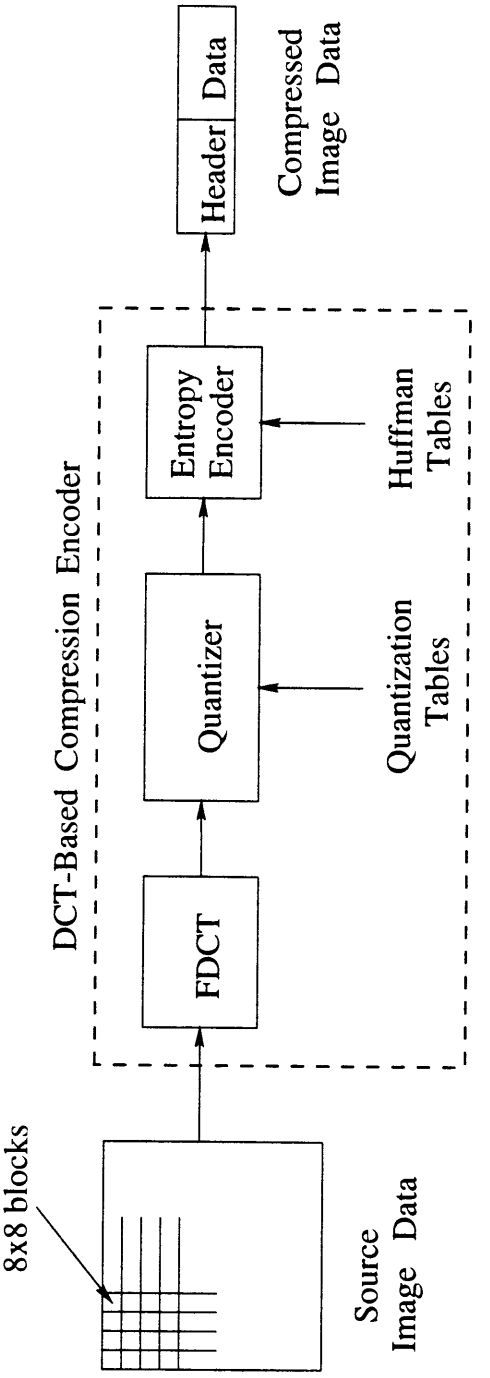


Figure 3.1: Simplified encoder block diagram

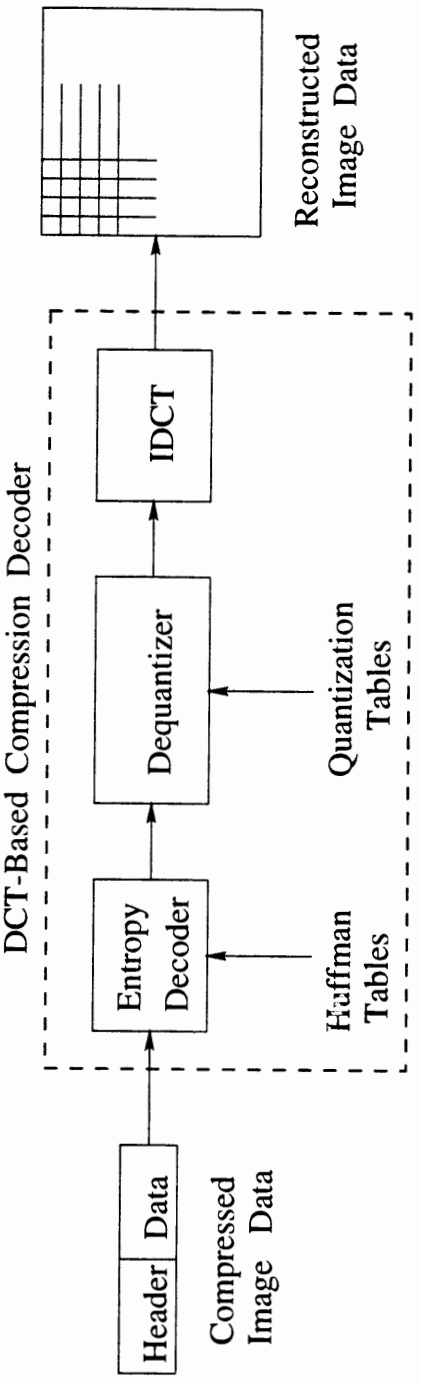


Figure 3.2: Simplified decoder block diagram

of gray levels. For the baseline algorithm, n is limited to 8 bits. So 128 is subtracted from each pixel value.

As an example if the input consists of an 8×8 block of the original image as in Matrix 3.1.

$$\begin{bmatrix} 139 & 144 & 149 & 153 & 155 & 155 & 155 & 155 \\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156 \\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156 \\ 159 & 161 & 162 & 160 & 160 & 159 & 159 & 159 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157 \\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157 \\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158 \end{bmatrix} \tag{3.1}$$

The original image has 256 or 2^8 possible gray levels, so the compression process begins by level shifting the pixels of the original subimage by -2^7 or -128 gray levels. The resulting shifted matrix is

$$\begin{bmatrix} 11 & 16 & 21 & 25 & 27 & 27 & 27 & 27 \\ 16 & 23 & 25 & 28 & 31 & 28 & 28 & 28 \\ 22 & 27 & 32 & 35 & 30 & 28 & 28 & 28 \\ 31 & 33 & 34 & 32 & 32 & 31 & 31 & 31 \\ 31 & 32 & 33 & 34 & 34 & 27 & 27 & 27 \\ 33 & 33 & 33 & 33 & 32 & 29 & 29 & 29 \\ 34 & 34 & 33 & 35 & 34 & 29 & 29 & 29 \\ 34 & 34 & 33 & 33 & 35 & 30 & 30 & 30 \end{bmatrix} \tag{3.2}$$

Then each shifted block data is transformed using the two-dimensional forward DCT independently. The JPEG forward DCT is defined slightly different from the standard DCT in the sense of scaling factor. The formula used in the JPEG standard is as follows:

$$F(u,v) = \frac{C(u)C(v)}{4} f(j,k) \sum_{j=0}^{7} \sum_{k=0}^{7} \cos\left[\frac{(2j+1)u\pi}{16}\right] \cos\left[\frac{(2k+1)v\pi}{16}\right] \tag{3.3}$$

$$C(w) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } w = 0 \\ 1 & \text{for } w = 1, 2, \ldots, n-1 \end{cases} \tag{3.4}$$

Where $f(j,k)$ represents the pixel value in the spatial domain and $F(u,v)$ represents the DCT coefficients value in the transform domain. After 2-D DCT the previous example has the transform coefficients shown below:

$$\begin{bmatrix} 235.6 & -1.0 & -12.1 & -5.2 & 2.1 & -1.7 & -2.7 & 1.3 \\ -22.6 & -17.5 & -6.2 & -3.2 & -2.9 & -0.1 & 0.4 & -1.2 \\ -10.9 & -9.3 & -1.6 & 1.5 & 0.2 & -0.9 & -0.6 & -0.1 \\ -7.1 & -1.9 & 0.2 & 1.5 & 0.9 & -0.1 & 0.0 & 0.3 \\ -0.6 & -0.8 & 1.5 & 1.6 & -0.1 & -0.7 & 0.6 & 1.3 \\ 1.8 & -0.2 & 1.6 & -0.3 & -0.8 & 1.5 & 1.0 & -1.0 \\ -1.3 & -0.4 & -0.3 & -1.5 & -0.5 & 1.7 & 1.1 & -0.8 \\ -2.6 & 1.6 & -3.8 & -1.8 & 1.9 & 1.2 & -0.6 & -0.4 \end{bmatrix} \tag{3.5}$$

The DCT coefficient matrix shows the spectral compression characteristics. The coefficient at the upper left-hand corner of the matrix is the "DC coefficient" which in our example is 235.6. This value represents the average of the overall magnitude

of the input matrix. We should note that the DC coefficient is significantly greater than the other coefficients in the DCT matrix, and also as the elements move farther and farther from the DC coefficient, they tend to become smaller and smaller. This means most of the useful information is concentrated in the upper left coefficients of the DCT matrix.

Quantization

The goal of quantization is to discard the information which is not visually significant. Quantization is defined as the DCT coefficients are scaled using a user-specified normalization array that is fixed for all blocks, followed by rounding off to the nearest integer:

$$F^*(u,v) = NINT\left(\frac{F(u,v)}{Q(u,v)}\right) \tag{3.6}$$

Where $F^*(u,v)$ and $Q(u,v)$ represent the quantized coefficient and normalization matrix element, respectively. $NINT$ stands for rounding operation. For the baseline system, four different normalization arrays are allowed (e.g., to accommodate luminance and chrominance components of an image). Each component of the normalization array is an 8-bit integer that in effect determines the quantization step size. The quality and bit rate of an encoded image can be varied by changing this array. The normalization matrix may be designed according to the perceptual importance of the DCT coefficients under the intended viewing conditions. A typical normalization array that has been used as a default quantization table by the JPEG standard is:

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{3.7}$$

After normalization/quantization process, the quantized coefficients of our example becomes:

$$\begin{bmatrix} 15 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.8}$$

We should notice that the normalization/quantization process typically results in many zero-valued coefficients which can be coded more efficiently.

Zigzag Scan

Since in the quantized DCT coefficients so many values are set to zero, JPEG standards use a zigzag pattern to scan the 2-D coefficients into a 1-D format. This zigzag pattern rearranges the coefficients in approximately decreasing order of their average energy (as well as approximately in order of increasing spatial frequency) with the aim of creating large runs of zero values. The zig-zag pattern moves from the left upper corner to the right lower corner along diagonal paths shown below (ordered from 0 to 63):

$$\begin{bmatrix} 0 & 1 & 5 & 6 & 14 & 15 & 27 & 28 \\ 2 & 4 & 7 & 13 & 16 & 26 & 29 & 42 \\ 3 & 8 & 12 & 17 & 25 & 30 & 41 & 43 \\ 9 & 11 & 18 & 24 & 31 & 40 & 44 & 53 \\ 10 & 19 & 23 & 32 & 39 & 45 & 52 & 54 \\ 20 & 22 & 33 & 38 & 46 & 51 & 55 & 60 \\ 21 & 34 & 37 & 47 & 50 & 56 & 59 & 61 \\ 35 & 36 & 48 & 49 & 57 & 58 & 62 & 63 \end{bmatrix} \tag{3.9}$$

For our example, this reordering results in:

$$15 \quad 0 \quad -2 \quad -1 \quad -1 \quad -1 \quad 0 \quad 0 \quad -1 \quad \text{All Zeros} \tag{3.10}$$

Entropy Coding

Since the one-dimensionally reordered array generated under the zigzag pattern is qualitatively arranged according to increasing spatial frequency, the JPEG coding

procedure is designed to take advantage of the long runs of zeros that normally result from the reordering. In particular, the DC coefficient is difference coded relative to the DC coefficient of the previous subimage or block. This difference is first assigned to one of the twelve categories shown in Table 3.1, where the values in category k are in the range $(2^{k-1}, 2^k - 1)$ or $(-2^k + 1, -2^{k-1})$, and $11 \geq k \geq 0$. A set of Huffman codes (base codes) with a maximum codeword length of 16 bits is used to specify the different categories. The JPEG default luminance DC code table is shown in Table 3.2. For each category, it is necessary to send an additional k bits to completely specify the sign and magnitude of a difference value within that category.

In our example, suppose the DC coefficient of the previous transformed and quantized block was 24, the resulting DPCM difference is $15 - 24$ or -9, which lies in category 4 of Table 3.1. In accordance with the default JPEG Huffman difference code of Table 3.2, the proper base code for a category 4 difference is 101 (a 3-bit code), while the total length of a completely encoded category 4 coefficient is 7 bits. The remaining 4 bits must be generated from the least significant bits (LSBs) of the difference value. For a general DC difference category (say, category K), an additional K bits are needed and computed as either the K LSBs of the positive difference or the K LSBs of the negative difference minus 1. For a difference of -9, the appropriate LSBs are $(0111) - 1$ or 0110, and the complete DPCM coded DC code word is 1010110.

To encode the AC coefficients, each nonzero coefficient is first described by a pair, Run/Category, where Category defines a category k for the coefficient amplitude in a way similar to the DC difference values and Run gives the position of the current

Table 3.1: DC and AC coefficient grouping

Category	Coefficient Range
0	0
1	-1,1
2	-3,-2,2,3
3	-7,...,-4,4,...,7
4	-15,...,-8,8,...,15
5	-31,...,-16,16,...,31
6	-63,...,-32,32,...,63
7	-127,...,-64,64,...,127
8	-255,...,-128,128,...,255
9	-511,...,-256,256,...,511
10	-1023,...,-512,512,...,1023
11	-2047,...,-1024,1024,...,2047

Table 3.2: JPEG default DC code (luminance)

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

coefficient relative to the previous nonzero coefficient, i.e., the runlength of zero coefficients between nonzero coefficients. The runlengths can range from 0 to 15, and a separate pair 15/0 (Run = 15 and Category = 0) represents a runlength of 16 zero coefficients. If the runlength exceeds 16 zero coefficients, it is coded by using multiple symbols. In addition, a special pair 0/0 is used to code the end of block (EOB) which signals that all the remaining coefficients in the block are 0s. Then each pair is Huffman coded similarly from the JPEG default table shown in Table 3.3 and followed by the additional bits required to specify the sign and exact amplitude of the coefficient in each of the categories.

If we choose the default JPEG AC code table, the first nonzero AC coefficient in our example (-2) is coded as 11100101. The first 6 bits of this code indicate that the coefficient was in category 2 and preceded by 1 zero-valued coefficient; the last 2 bits are generated by the same process used to arrive at the LSBs of the DC

difference code. Continuing in this manner, the completely coded block data is

1010110/11100101/000/000/000/110110/1010

3.2.2 Decoding

Entropy decoding process is the first step performed in the receiver. When the transmitted stream of data is received, it can easily be decoded by the Huffman code table at the receiver which is exactly the same as that in the transmitter. Because a Huffman coded binary sequence is instantaneous and uniquely decodable, this step is easily accomplished in a simple lookup table manner. Here the regenerated array of quantized coefficients is

$$\begin{bmatrix} 15 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.11}$$

The quantized coefficients are then dequantized by multiplying the normalization matrix, the array becomes

Table 3.3: JPEG default AC code example (luminance)

Zero Run	Category	Codelength	Codeword
0	1	2	00
0	2	2	01
0	3	3	100
0	4	4	1011
0	5	5	11010
0	6	6	111000
0	7	7	1111000
.	.	.	.
1	1	4	1100
1	2	6	111001
1	3	7	1111001
1	4	9	111110110
.	.	.	.
2	1	5	11011
2	2	8	11111000
.	.	.	.
3	1	6	111010
3	2	9	111110111
.	.	.	.

Table 3.3 continued

Zero Run	Category	Codelength	Codeword
4	1	6	111011
5	1	7	1111010
6	1	7	1111011
7	1	8	11111001
8	1	8	11111010
9	1	9	111111000
10	1	9	111111001
11	1	9	111111010
.	.	.	.
End of Block (EOB)		4	1010

$$\begin{bmatrix} 240 & 0 & -10 & 0 & 0 & 0 & 0 & 0 \\ -24 & -12 & 0 & 0 & 0 & 0 & 0 & 0 \\ -14 & -13 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.12}$$

The dequantized coefficients are then inverse transformed using an integer DCT according to

$$\hat{f}(j,k) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v)\hat{F}(u,v) \cos\left[\frac{(2j+1)u\pi}{16}\right] \cos\left[\frac{(2k+1)v\pi}{16}\right] \tag{3.13}$$

$$C(w) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } w = 0 \\ 1 & \text{otherwise} \end{cases} \tag{3.14}$$

The value 128 is added back to each pixel value to yield

$$\begin{bmatrix} 144 & 146 & 149 & 152 & 154 & 156 & 156 & 156 \\ 148 & 150 & 152 & 154 & 156 & 156 & 156 & 156 \\ 155 & 156 & 157 & 158 & 158 & 157 & 156 & 155 \\ 160 & 161 & 161 & 162 & 161 & 159 & 157 & 155 \\ 163 & 163 & 164 & 163 & 162 & 160 & 158 & 156 \\ 163 & 163 & 164 & 164 & 162 & 160 & 158 & 157 \\ 160 & 161 & 162 & 162 & 162 & 161 & 159 & 158 \\ 158 & 159 & 161 & 161 & 162 & 161 & 159 & 158 \end{bmatrix} \tag{3.15}$$

The difference between the original and reconstructed subimage are a result of the lossy nature of the JPEG compression and decompression process. In this example, the errors range from -5 to $+5$ and are distributed as follows:

$$\begin{bmatrix} -5 & -2 & 0 & 1 & 1 & -1 & -1 & -1 \\ -4 & 1 & 1 & 2 & 3 & 0 & 0 & 0 \\ -5 & -1 & 3 & 5 & 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & -2 & -1 & 0 & 2 & 4 \\ -4 & -3 & -3 & -1 & 0 & -5 & -3 & -1 \\ -2 & -2 & -3 & -3 & -2 & -3 & -1 & 0 \\ 2 & 1 & -1 & 1 & 0 & -4 & -2 & -1 \\ 4 & 3 & 0 & 0 & 1 & -3 & -1 & 0 \end{bmatrix} \tag{3.16}$$

The root mean-squared error (RMSE) is defined as

$$RMSE = \sqrt{\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}[f(i,j) - \hat{f}(i,j)]^2} \tag{3.17}$$

N and M are the width and height, respectively, of the images in pixels, f is the original image, and $\hat{f}$ is the reconstructed image. For this specific block, the root mean-squared error between the original image block and the reconstructed image block is

$$RMSE = \sqrt{\frac{1}{64}\sum_{i=1}^{7}\sum_{j=1}^{7}[f(i,j) - \hat{f}(i,j)]^2} = 2.26 \tag{3.18}$$

Chapter 4

Blocking and Ringing Artifacts in JPEG Coded Images

4.1 Introduction

In this chapter, the well-known JPEG coding artifacts are first introduced. Then some existing artifact removal techniques are briefly discussed.

4.2 JPEG Lossy Image Compression Artifacts

The baseline JPEG DCT is a lossy image compression technique. The image quality can be traded for lower bit rate by scaling the normalization matrix components. Larger values correspond to coarser quantization, lower bit rate, and lower reconstructed image quality. At low bit rates, some well-known artifacts arise. The two most obvious artifacts in a low bit coded image are "blocking" and "ringing" effects. JPEG block-based DCT causes discontinuities between adjacent blocks which is known as *blocking* effect. And coarse quantization discards the high frequency DCT coefficients and thus causes contouring along sharp edges on the uniform background which is called *ringing* effect. These two kinds of artifacts can be seen from

Fig. 4.2, which is the JPEG encoded "Lena" picture at 0.25 bits/pixel (bpp). The original "Lena" picture is also given in Fig. 4.1 for comparison purpose.

Figure 4.1: The Original "Lena" Picture

Figure 4.2: The JPEG encoded "Lena" Picture at 0.25 bpp

4.3 Different Approaches To Reduce Compression Artifacts

In the past two decades, a variety of efforts have been made to reduce blocking and ringing artifacts, primarily in two major categories: choosing different coding schemes at the encoding end and post-processing at the decoding end.

4.3.1 Different Encoding Schemes

One simple solution at the encoder site is to devide an image into blocks with a slight overlap. The one-pixel overlap scheme was reported to be the most effective one in [1]. Although the overlap method alleviates blocking effects well without degrading image content, the major disadvantage of the method is a 13% increase in bit rate and change of the coding procedure.

Lynch et al. [2] proposed an Edge Compensated Transform Coding (ECTC) scheme to preprocess the image before transform coding. They tried to use an edge detector to mask out the blocks which exist ringing artifact. These blocks are encoded with side information which is different from the JPEG standard. The other blocks still use the JPEG transform coding. At the receiver, these two kinds of blocks are decoded separately, and an image assembler put them together to get the reconstructed image. ECTC images are superior to the JPEG standard images subjectively, but the standard compression outperforms ECTC objectively. Again, this method has the same disadvantage as [1] in changing the coding procedure.

Recently, another blocking effect removal coding strategy was presented by *Luo et al.* [3]. They believe that a good result in blocking effect removal can not be achieved if one concentrates only on the block boundary area. The DC components serves as the reference to all the pixels within a block, the calibration or adjustment of the coded DC components is necessary before the application of smoothing techiques. They choose two blocks centered in the image as the anchor blocks and their DC components are coded error free. In order to use these already DC cali-

brated blocks to maximize the neighborhood constraints, they designed a novel scan scheme which starts scanning from the two centered anchor blocks and traverse the entire image in two spiral paths shown in Fig. 4.3.

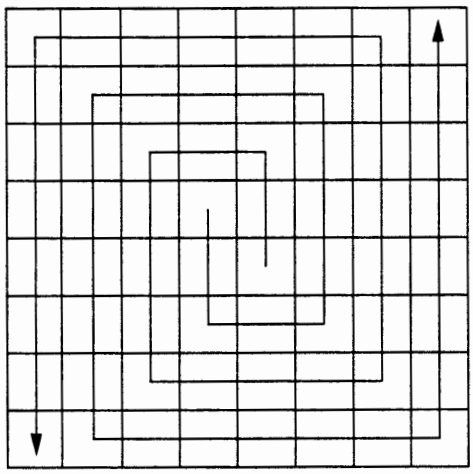


Figure 4.3: Spiral Scanning Routes

The authors reported the above scanning scheme is more effective than the JPEG accepted one. But abandoning the existing standard restricts its application.

4.3.2 Post-processing Algorithms

Unlike the change of encoding schemes, post-processing algorithms used at popular image compression decoders are particulary interesting because they do not require that the bit stream be altered. Thus, they allow a decoder to have a great advantage while remaining compatible with the existing encoders.

In the paper [1], *Reeve* and *Lim* also proposed a filtering method. In this method,

a simple low pass filter is applied to the pixels that are directly adjacent to the block boundaries. The filter they chose is a 3×3 Gaussian low pass filter shown in Fig. 4.4.

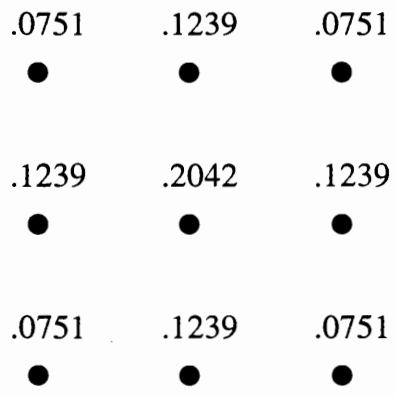


Figure 4.4: A 3×3 Gaussian Low Pass Filter

The filtering method can reduced blockiness, but it also blurred images (especially the actual image detail at the block boundaries).

Lynch et al. in another paper [4] proposed a post-processing algorithm to apply space varying filtering in low frequency blocks and edge blocks. Low frequency blocks are identified in the transform coefficient domain; edge blocks are identified in the spatial domain. A block is declared as low frequency block if:

$$C_{DCT}(i,j) * K_{low} = \hat{0} \tag{4.1}$$

$C_{DCT}(i,j)$ is the 8×8 block of quantized DCT coefficients of block (i,j). $*$ is element by element multiplication. K_{low} is the test matrix, and $\hat{0}$ is the 8×8 0 matrix. The test matrix used is zero everywhere except in a 2×2 square at the top left corner where it is one. Edge blocks are detected by first classifying the pixels

into two types: non-edge pixel and edge pixel. Then the distribution of these two types of pixels determines the block property.

After the block detection step, a low-pass filter with kernel size from 3×3 to 9×9 applies to the low frequency blocks and the flat regions in the edge blocks. The choice of the filter kernel size depends on the size of the flat region.

The simulation result presented by the authors shows this post-processing scheme can improve the SNR about 0.1 dB at the high bit rate and 0.4 dB at the low bit rate. The objective performance is also satisfied.

An alternative adaptive postprocessor was employed by *Kuo et al.* [5]. In their approach, a visibility factor V is defined as the summation of the absolute pixel value difference between the block and its four neighboring blocks. After obtaining factor V for each block in an image, the mean $E(V)$ and the standard deviation $SD(V)$ are calculated based on all the visibilities. Then a threshold value V_t is chosen as $V_t = E(V) + SD(V)$. The blocks with visible blocking effects are identified if their V values are greater than the threshod V_t. The Robinson gradient operator is used to find edges. False edges are removed by so called *edge tree structure*. Finally, a space-variant lowpass filter is applied to the blocks with visible blocking effects and the flat regions along the edges. Because its space-variant property, the edge pixels will not be touched. The filter they choose is 3×3 lowpass filter with coefficient $3/11$ at the center and $1/11$ at the other places. A filter coefficient is reset to zero if it masks a pixel at the edge or the other side of the edge in order not to smooth the edge and take image pixel at the other region into consideration. In this case, the summation of the filter coefficients have to be renormalized to one to ensure the

average of the processed image is unchanged.

The above post-processing algorithms are based on spatial filtering. Another approach for reducing coding artifacts is to use image restoration theory.

Zakhor [6] proposed an iterative blocking effect reduction technique based on the theory of projection onto convex sets (POCS). The basic idea behind this technique is to impose a number of constraints on the block DCT coded image to restore it to its artifact-free form. One such constraint is that the block transform coded image suffering from blocking effects contains high-frequency vertical and horizontal components. The artifact-free image should be bandlimited. Thus one step in the iterative procedure is to apply a low-pass filter to the image. The filter he chose is a 3×3 finite impulse reponse (FIR) filter of the form

$$h(0,0) = 0.2042$$

$$h(0,1) = h(0,-1) = h(1,0) = h(-1,0) = 0.1239$$

$$h(0,2) = h(0,-2) = h(2,0) = h(-2,0) = 0.0751$$

Another constraint is related to the quantization intervals of the transform coefficients. Specifically, the decision levels associated with transform coefficient quantizers can be used as lower and upper bounds on transform coefficients. This is the boundaries of the convex set for projection. The out-of-bound coefficient will be set to the upper bound of the quantization interval if its value is greater than the upper bound. Otherwise, it will be set to the lower bound if its value is less than the lower bound. The iterative procedure is proceeded by applying these two

constraints alternatively until the pre-selected the convergence condition is satisfied. Usually, 5 iterations are needed to get a image without noticeable blocking artifact and excessively blurry. The algorithm converges after 20 iterations or so.

Yang et al. in [7] proposed a spatially adaptive smoothness constraint to further improve the performance of POCS method. The key point they captured is that the blocking artifact is not equally noticeable through the whole image. It is more visible in smooth regions than in texture or edge areas due to the human perceptual characteristics. Instead of using the same filter for the whole image as mentioned in [6], they apply a spatially-adaptive smoothness filter to the image. The weights of the filter can be changed locally according to the local statistics of the image. This spatially adaptive smoothness constraint set resulted in better images, both visually and objectively, than the previous POCS space-invariant algorithm in [6]. The penalty for the improvement in image quality is an increase in the complexity of the decoder.

Later in [8], *Su et al.* have developed another POCS algorithm to reduce both blocking and ringing artifacts in block transform-coded images. A new edge preservation constraint set was presented in their paper. Before applying this new constraint set to the decompressed image, they labeled each pixel as edge, texture, or uniform. Also a fourth pixel type *coastal* is defined as a non-edge pixel that has an edge pixel among its eight nearest neighboring pixels. The classificaion of the pixels are based on the computation of the local (3×3) variance at each pixel location. Then each 8×8 block is labeled as type of uniform, uniform/texture, texture, edge/texture, medium edge, or strong edge according to the numbers of uniform

pixel and edge pixel in the block. The change of pixel value for every location in the decompressed image is restricted in a range obtained from the pixel type at that location and the block type it belongs to. This algorithm still has the problem of the computation complexity because of its iteration nature.

Chapter 5

A Novel Post-processing Algorithm

5.1 Introduction

In this chapter, a novel post-processing algorithm is proposed and explained step by step.

5.2 Post-processor Design Overview

For the block-transform encoded image, the blocking artifact appears not only on the block boundaries but also in their neighborhood. It may improve the PSNR by simply smoothing the block boundaries, but it can not reduce the blocking artifact as much as we want. And the ringing artifact appears along sharp edges in the blocks. It is an important step to detect these edges for ringing artifact reduction. Our proposed algorithm include these main steps: block classification, blocking artifact removal, ringing artifact removal, and fidelity constraint. In block classification step, we classify blocks with blocking artifacts and ringing artifacts in transform domain. For the blocking artifact removal, we apply a simple FIR filter across the

block boundaries and use mid-point displacement interpolation for the macro-blocks which have blocking artifacts. For the ringing artifact removal, we first detect the edges in spatial domain, then merge the regions along the edges and use a low-pass filter in these regions. Finally, quantization table and original received DCT coefficients are being used to meet the fidelity constraint. The block diagram is shown in Fig. 5.1 and algorithm details are described in the following sections.

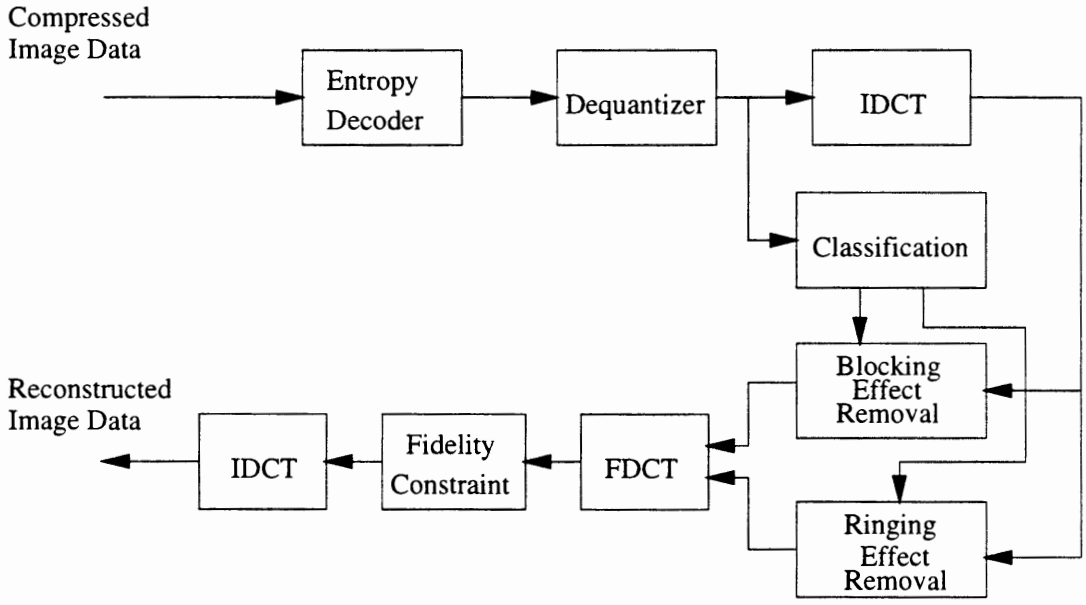


Figure 5.1: The block diagram of the proposed postprocessor

5.3 Block Classification

We know that blocking and ringing effects do not appear significantly in every block of a coded image at low bit rate. This can be seen from Fig. 4.2, which is the JPEG encoded "Lena" at 0.25 bits/pixel (bpp).

In order to remove blocking and ringing effects, first, we have to detect the areas which have blocking or ringing artifacts. It is clear that blocking artifacts are more visible in the low frequency blocks and ringing artifacts show up along the sharp edges, in other words, in the high frequency blocks. In our approach, we use the method similar to [4] to classify the low frequency and high frequency blocks in the transform coefficient domain. A block is marked as low frequency block if:

$$C_{DCT}(i,j) * K_{low} = \hat{0} \tag{5.1}$$

Similarly, a block is marked as high frequency block if:

$$C_{DCT}(i,j) * K_{high} \neq \hat{0} \tag{5.2}$$

where C_{DCT} is the 8×8 block of quantized DCT coefficients of block (i,j). $*$ is element-by-element multiplication. K_{low}, K_{high} are the test matrices for detection of low frequency block and high frequency block, respectively. And $\hat{0}$ is the 8×8 matrix of zeros.

After a series of experiments, we choose

$$K_{low} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{5.3}$$

and

$$K_{high} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{5.4}$$

With this choice, we can locate the blocks with visible blocking effects and ringing effects in the JPEG encoded pictures at low bit rate. Fig. 5.2 and Fig. 5.3 shows the location of blocks with blocking artifacts and ringing artifacts, respectively. The blocking and ringing blocks have be marked off from the corresponding figures.

From the results we got from different test images, the above technique works well and has less calculation. The following post-processing steps can rely on it.

5.4 Removal of Blocking Artifacts

5.4.1 Block Boundary Filtering

According to our observations, the pixels on the block boundaries exist discontinuity whether this block belongs to the blocking block or not. In order to get less MSE, we can apply block boundary filtering for all the blocks in the picture. On the other hand, we have less calculation if we only smooth the blocking block

Figure 5.2: The blocks with blocking artifact

boundaries. The performance of the latter is only slight worse than the previous one.

The discontinuity at the block boundary can be simply lowpass filtered. The mask for this filter [5] is shown in Fig. 5.4. For the pixels at block boundaries but not at corners, the 2×1 mask with filter coefficients 0.75 and 0.25 is used. As for

Figure 5.3: The blocks with ringing artifact

the pixels at corners of the block, a 2×2 mask with filter coefficients 0.5, 0.25, 0.25, and 0 is used. With this simple space-variant lowpass filter, the discontinuity between blocks can be reduced.

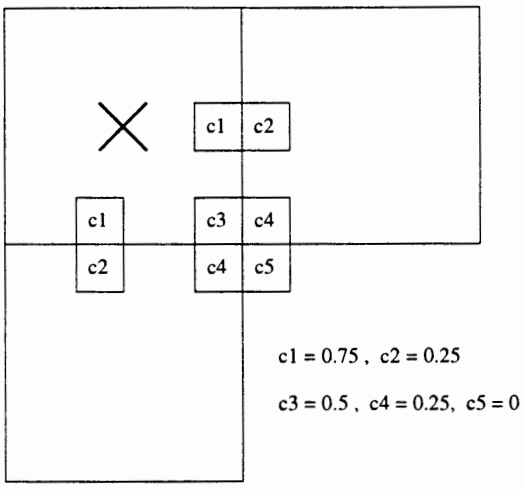


Figure 5.4: The low-pass filter for the pixels at the block boundary. X is the block being processed

5.4.2 Mid-point Displacement Interpolation

Block boundary filtering can only reduce the blocking effects across the boundaries and cannot touch the pixels inside the blocks. In fact, a good result in block effect removal can not be achieved if we only deal with the pixels on the block boundary area. The reason for this is that the DC coefficient in the transform domain serves as the reference to all the pixels within a block, including the pixels along the block boundaries. In a large flat area, the difference of DC coefficients from adjacent blocks can cause severe blocking effects which are not limited on the block boundary area. To solve this problem, a DC calibration algorithm is proposed in [3]. But their approach has to change the JPEG encode scheme, thus it can not be used in existing JPEG standard.

To keep the post-processor compatible with the universal accepted JPEG standard, we choose mid-point displacement interpolation as our tool to remove the

blocking effect appeared in a large gradual change region. Before we do mid-point displacement interpolation, we have to specify the areas to be processed. The rule we set is: if the four adjacent blocks connected each other all have be classified as blocking block from the image segmentation step, then these four blocks combine to be a macro block and need to be applied mid-point displacement interpolation.

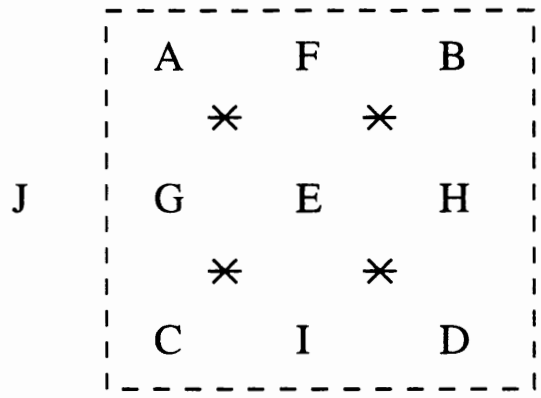


Figure 5.5: The Mid-point Displacement Interpolation

Next, we apply mid-point displacement interpolation on these macro blocks. For a specific macro block being concerned, we choose a center pixel from each of the four blocks, say at the location $(4,4)$, as our starting point. These center pixels are denoted as A, B, C and D in Fig. 5.5, respectively. Then the pixel at location E which has equal distance with A, B, C and D will be interpolated. The new pixel value at location E is generated by taking an average of the surrounding pixels A, B, C and D. The mid-point displacement algorithm is a recursive algorithm. On the second stage, we will fill in the pixels at the location F, G, H and I. The pixel value at these locations has two choices. Let the pixel at location G be an example.

It is the center of pixel A, E, C and J. And the pixel J does not belong to the macro block being focused on. If the block which the pixel J "sits" in is a blocking block, then the pixel value of G will be the average of pixels A, E, C, and J. Otherwise, the pixel G keeps its original value. The same rule is valid for the pixels F, H and I. On the next stage, all the pixels at the location "*" will be interpolated. This process continues until all the pixels have been filled.

Up to now, the blocking artifacts have been removed. We will move to ringing artifact reduction section.

5.5 Removal of Ringing Artifact

Since the ringing blocks have been found, the next step is to remove ringing artifacts in these ringing blocks.

Typically, a ringing block can be modeled as several smooth regions separated by edges. Thus the first step of the ringing reduction algorithm is edge detection. The necessity of edge detection is to conserve the edges while applying an edge-adaptive lowpass filter within each of the smooth regions.

There are many gradient operators available for edge detection. Sobel operator is our choice because it is simple and easy to be implemented in digital hardware. For Sobel edge detection, the pixel at location (m, n) is declared as an edge pixel if $g_{H1}(m, n)$ or $g_{H2}(m, n)$ is greater than a chosen threshold t, where $g_{H1}(m, n)$ and $g_{H2}(m, n)$ are the outputs of the filters whose impulse functions are the Sobel masks.

The Sobel masks $H1$ and $H2$ are defined as:

$$H1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{5.5}$$

$$H2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{5.6}$$

The threshold t is determined by experiments in order to get the least mean square error (MSE) of the post-processed image. We found the threshold t equal to 15 is the best one for the "Lena" picture. For the other test images, the best choice of t is around 15. If we don't want to change the threshold while the image changes, we can set $t = 15$ as our default value. Although we cannot get the best result, the actual result is very close to it.

As a result of edge detection, each ringing block is divided into several small regions separated by the edges. Fig. 5.6 shows an ringing block example. The edge pixels are marked with a black dot. The pixels are scanned in one by one from left to right and from top to bottom. First, Pixel a_1 is checked. It is a non-edge pixel and belongs to Region a. Then the pixel just right of a_1 will be checked. This time it is an edge pixel and will not be assigned to any smooth regions. Next Pixel f_1 will be identified as a non-edge pixel and is separated from Pixel a_1 by an edge pixel. At this moment it is believed that Pixel f_1 and a_1 are in different regions. A new region name f is given to Pixel f_1.

The rule to assign a non-edge pixel to an existing region is: the pixel to be considered is adjacent to a pixel within that region; the pixel value is within a predefined range, for example, half of the edge detection threshold t, around the average value of the existing pixels in that region. In this way, the value of Pixel f_2 is read in and compared with the value of f_1. Assuming their pixel values are close each other within the predefined range, they are considered in the same region f. When the value of Pixel f_3 is read in, it will be compared with the average value of f_1 and f_2. In our example, these two values are very close and Pixel f_3 is in the same region with f_1 and f_2. If not, Pixel f_3 will be in a different region like e_1 showed in the same figure. Similarly, Pixel a_1 through a_6 are declared in the same region a.

When the pixel marked with a cross is met, its value will be compared with the average value of region a and f individually. If its value is close to the average value of one region and within the predefined range, then it belongs to that region. In this case, the average value of region a and f will be evaluated next. If the difference is within the predefined range, region a and f can be merged into one region. On the other hand, if that pixel value is not close to either the average value of region a or f, the merge of regions will not happen.

This segmentation and merging process continues until all the non-edge pixels have been touched. In the example shown in Fig. 5.6, the non-edge pixels are segmented into 5 or 6 regions depending on the pixel value.

In the next step, smoothing is performed within each merged region. A simple low-pass filter which takes average value of the pixels in the region applies each

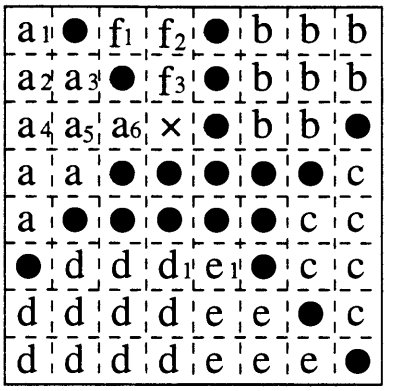


Figure 5.6: A Ringing Block Example

region individually. Since this low-pass filter does not touch the edge pixels, the edge will not be blurred by the filter. However, we may oversmooth the non-edge pixels. The following fidelity constraint section will discuss the way to overcome this shortcoming.

5.6 Fidelity Constraint

With the post-processing steps above, blocking and ringing artifacts have been minimized. But both blocking and ringing artifacts reduction have been done in the spatial domain. The transform domain constraint condition has not been considered. According to quantization theory, given quantization table (QT) and quantized DCT coefficients, the ranges of the original image's unquantized DCT coefficients have been confined. For example, if the quantizer stepsize is 16 and the quantized DCT value is 3, then the unquantized DCT value is in [40,55). If the decompressed image is perfectly recovered and free of blocking and ringing artifacts, its quantized DCT

coefficients should be the same with the original received DCT coefficients. This condition is often used in convex projection image recovery algorithms [8]. We will use this constraint to improve the bitstream consistency.

Let us start with blocking block constraint. After mid-point displacement interpolation, we perform DCT and quantization to these smoothed blocks. If the resulting unquantized DCT coefficients are within received values plus or minus quantization bin value (half of the quantizer stepsize), then the recovered DCT coefficients will be used. Otherwise, if an individual resulting quantized DCT coefficient is beyond the range, it will be replaced by the maximum or minimum value allowed in that range. For the previous example, the quantizer stepsize is 16 and quantized DCT value is 3. When the recovered DCT coefficient is larger than 40 and less than 55, this recovered DCT coefficient will be used. Otherwise, if it is larger than 55 or less than 40, 55 or 40 will be used accordingly.

The above step can overcome the over-smoothing problem in the blocking blocks. Similarly, we perform the same constraint to the ringing blocks. By doing this, we can avoid over-smoothing the ringing blocks due to the false regions caused by incorrect edge detection.

Chapter 6

Computer Simulation and Performance Comparison

6.1 Introduction

In this chapter, the performance of the proposed postprocessor is evaluated both objectively and subjectively by computer simulation. Two performance measures will be used here. One is root mean-squared error (RMSE). The other is peak signal-to-noise ratio (PSNR). PSNR in decibels (dB) is computed as

$$PSNR = 20 \log_{10} \frac{255}{RMSE}$$

where RMSE is defined as

$$RMSE = \sqrt{\frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} [f(i,j) - \hat{f}(i,j)]^2}$$

and N and M are the width and height, respectively, of the images in pixels, f is the original image, and $\hat{f}$ is the reconstructed image.

6.2 Computer Simulation Algorithm

In order to verify the performance of the proposed postprocessor, a test program is written in C language and run under UNIX platform. The implementation

algorithm is as follows:

1. Read in the original uncompressed image.
2. Read in the quantization table and specify the scaling factor.
3. Use JPEG baseline algorithm to compress the original image.
4. Use JPEG baseline algorithm to decompress the image.
5. In the DCT domain, Equations 5.3 and 5.4 are used to detect the blocks with blocking artifact and ringing artifact.
6. In the spatial domain, apply the low-pass filter shown in Fig. 5.4 to the block boundaries.
7. In the spatial domain, mid-point interpolation is applied to the macro blocks.
8. For the ringing blocks, Sobel edge detector is used to detect the edges within the block and low-pass filter is applied to the regions along the edges.
9. Take forward DCT again for the processed image and quantization constraint is used to adjust the DCT coefficients.
10. Take inverse DCT for the modified DCT coefficients to get the final post-processed image.
11. Calculate the root mean-squared error.

6.3 Simulation Results

The three 512×512-pixel test images, "Bar", "Lena" and "Peppers", are used in simulation experiments. They are both encoded at 0.25 bpp by using JPEG baseline algorithm and JPEG default quantization table. The main artifact in "Bar" image is ringing artifact. The "Lena" image has a lot of blocking artifact. Both blocking artifact and ringing artifact appear in the "Peppers" image.

The proposed postprocessor is applied to these JPEG encoded images. The performance measures in RMSE and PSNR are shown in Table 6.1. As expected, our proposed postprocessor achieved at least 0.6 dB improvement of PSNR over the standard JPEG compressed image at 0.25 bpp for the test images. This result shows the objective performance of this post-processing scheme is very significant.

For the image processing algorithm, the subjective performance is more important. Fig. 6.1 through Fig. 6.9 show the detail comparison between standard JPEG encoded images and our reconstructed ones. The subjective results are also dramatic. Note that the blocking artifacts in the shoulder and face areas of "Lena" picture have been removed. The ringing artifacts along the lines of "Bar" picture are almost invisible after post-processing. Similar results are seen in "Peppers" picture.

The experiment results have proved the success of the proposed algorithm.

Table 6.1: The Performance Comparison Between the Standard JPEG Encoded Images and Postprocessed Images at 0.25 bpp

image	type	RMSE	PSNR
Bar	JPEG	4.4620	35.14
	Proposed	3.1117	38.27
Lena	JPEG	7.7003	30.40
	Proposed	7.1646	31.03
Peppers	JPEG	7.9456	30.13
	Proposed	7.3646	30.79

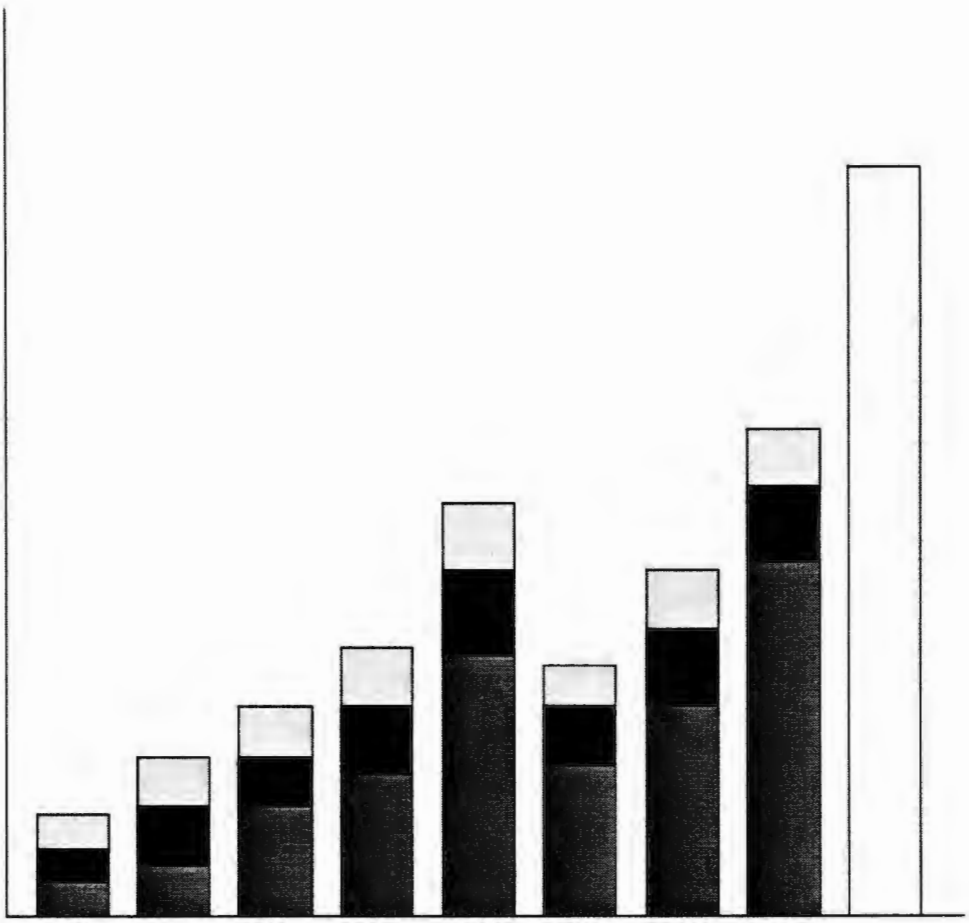


Figure 6.1: "Bar" picture - original

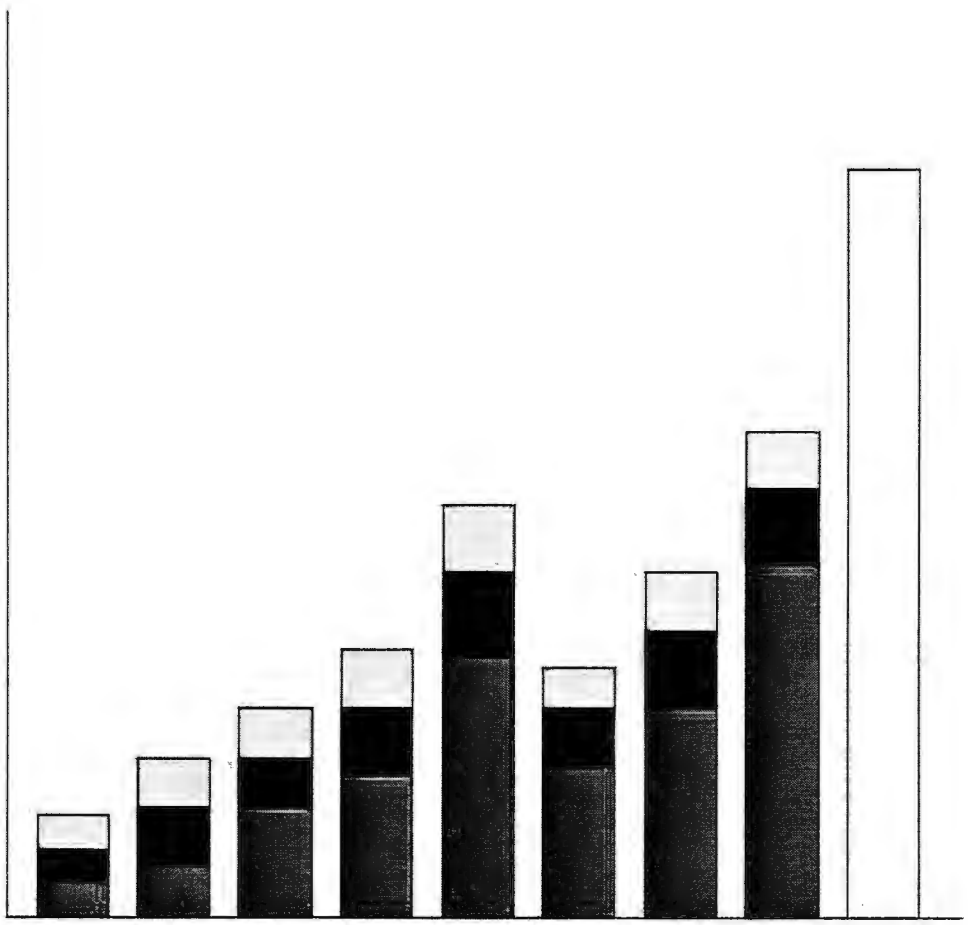


Figure 6.2: "Bar" picture - before post-processing - 0.25bpp

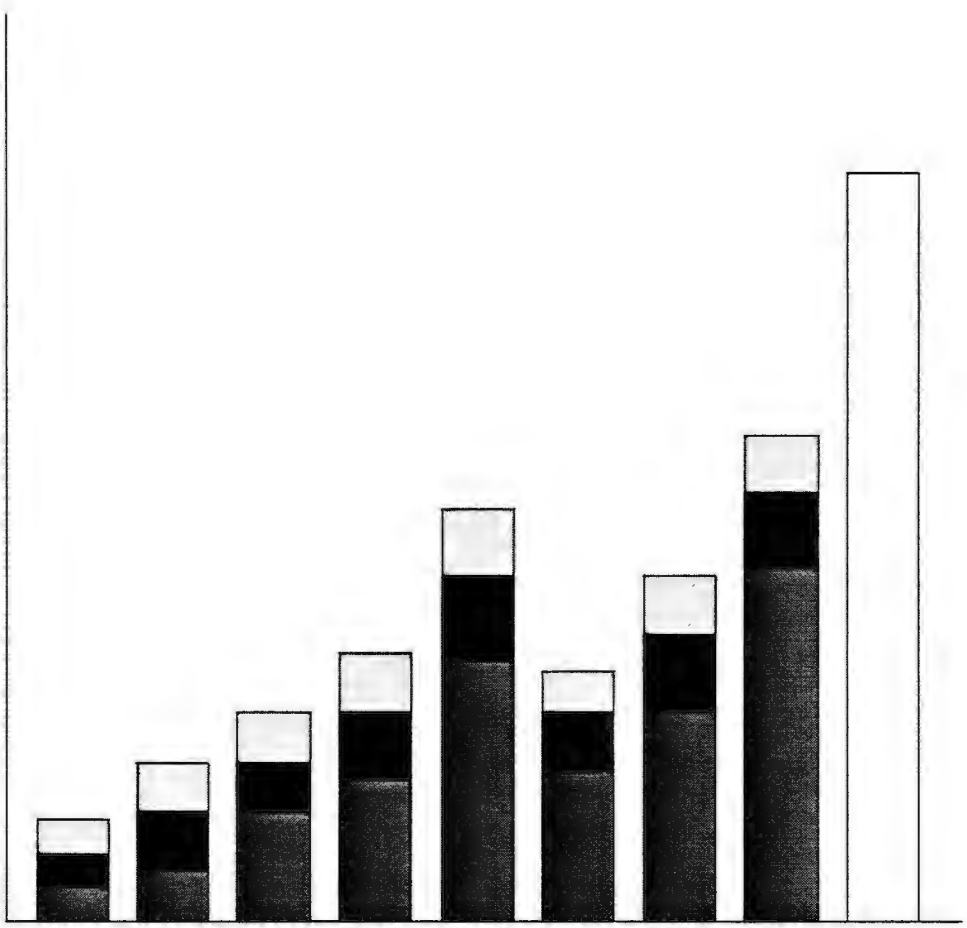


Figure 6.3: "Bar" picture – after post-processing - 0.25bpp

Figure 6.4: "Lena" picture - original

Figure 6.5: "Lena" picture - before post-processing - 0.25bpp

Figure 6.6: "Lena" picture - after post-processing - 0.25bpp

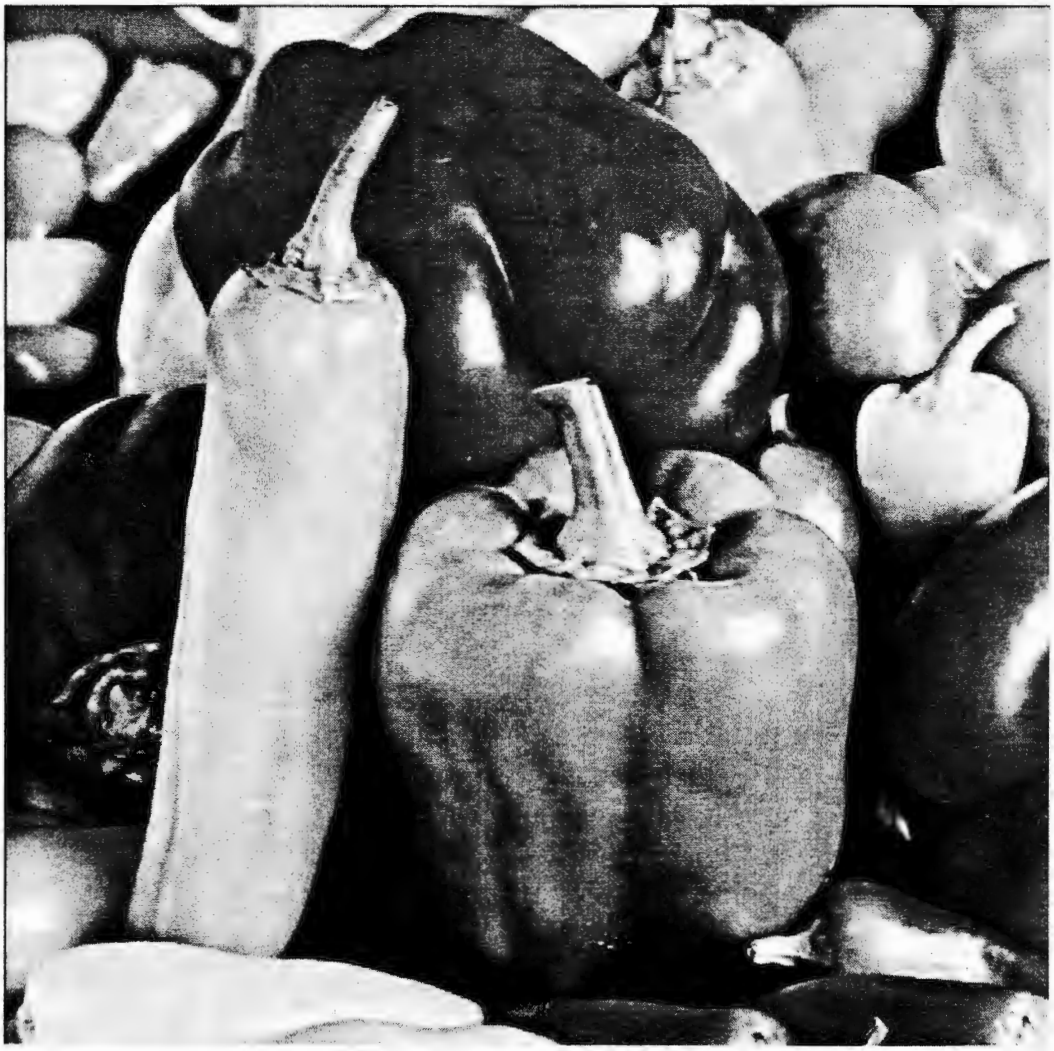


Figure 6.7: "Peppers" picture - original

Figure 6.8: "Peppers" picture - before post-processing - 0.25bpp

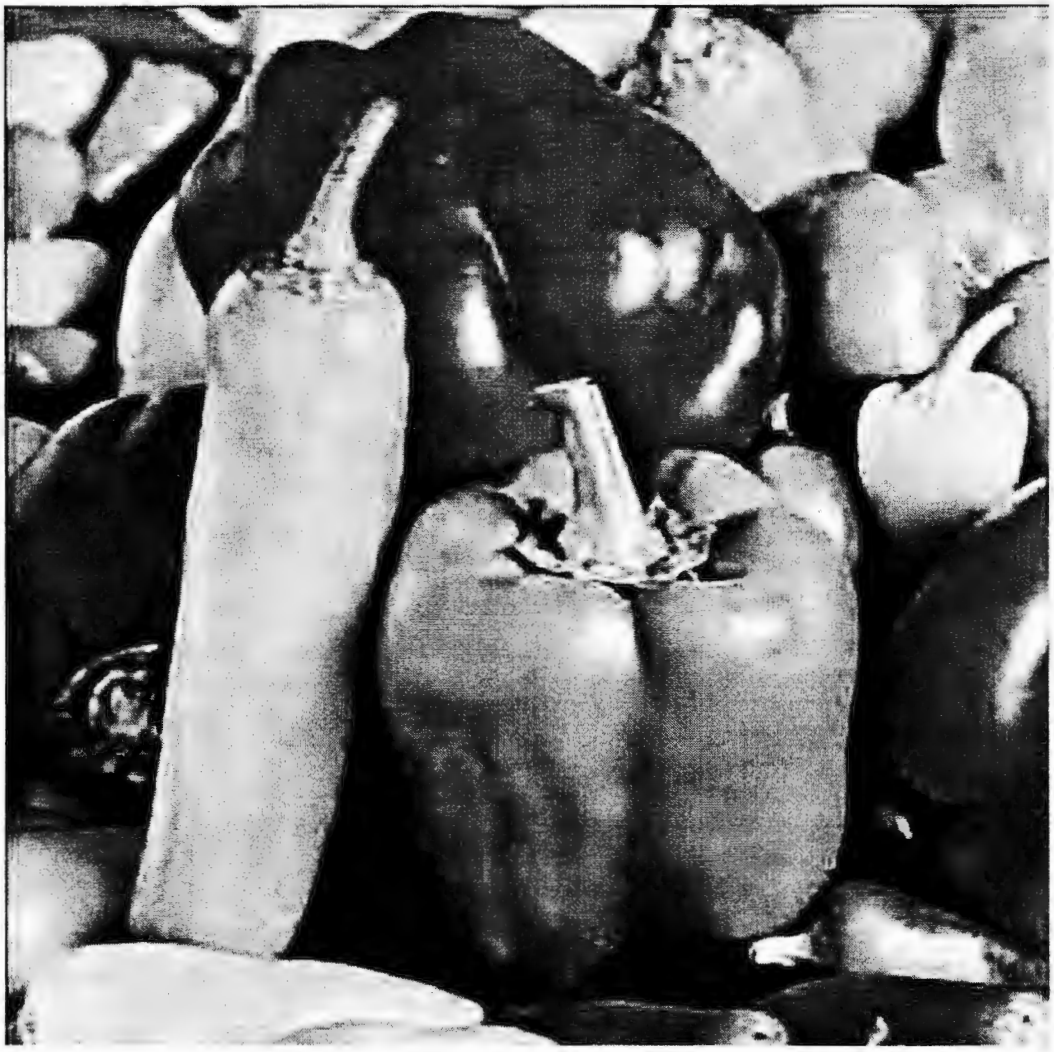


Figure 6.9: "Peppers" picture - after post-processing - 0.25bpp

Chapter 7

Conclusion

In this thesis, the general image compression techniques were discussed first, then JPEG baseline image compression algorithm was given in much detail, and the blocking and ringing artifacts were shown at low bit rate, followed by the review of existing methods to remove these artifacts.

The emphasis of the thesis was the presentation of a novel postprocessor to remove blocking and ringing artifacts for block transform encoded images. Both objective and subjective results showed the superiority of the new method. The proposed technique can be applied to the video intra-coded frames such as MPEG I frame. The extension of this algorithm to the video inter-coded frames will be the future work.

Bibliography

[1] H. Reeve and J. Lim, "Reduction of blocking effects in image coding," *Optical Engineering*, vol. 23(1), pp. 34–37, January/February 1984.

[2] W. E. Lynch, A. R. Reibman, and B. Liu, "Edge compensated transform coding," in *Proc. ICIP-94*, pp. 105–109, November 1994.

[3] J. Luo, C. W. Chen, K. J. Parker, and T. S. Huang, "A new method for block effect removal in low bit-rate image compression," in *Proc. ICASSP-94*, vol. V, pp. 341–344, April 1994.

[4] W. E. Lynch, A. R. Reibman, and B. Liu, "Post processing transform coded images using edges," in *Proc. ICASSP-95*, pp. 2323–2326, May 1995.

[5] C. J. Kuo and R. J. Hsieh, "Adaptive postprocessor for block encoded images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, pp. 298–304, August 1995.

[6] A. Zakhor, "Iterative procedures for reduction of blocking effects in transform image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, pp. 91–95, March 1992.

[7] Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos, "Projection-based spatially adaptive reconstruction of block-transform compressed images," *IEEE Transactions on Image Processing*, vol. 4, pp. 896–908, July 1995.

[8] J. K. Su and R. M. Mersereau, "Post-processing for artifact reduction in JPEG-compressed images," in *Proc. ICASSP-95*, pp. 2363–2366, May 1995.

[9] R. Stevenson, "Reduction of coding artifacts in transform image coding," in *Proc. ICASSP-93*, vol. V, pp. 401–404, April 1993.

[10] ISO/IEO CD 10918-1, "Digital compression and coding of continuous-tone still images, part 1: Requirement and guidelines," March 15 1991.

[11] ISO/IEC DIS 11172, "Coding of moving pictures and associated audio for digital storage media up to about 1.5 mbits/s," 1992.

[12] G. Wallace, "The JPEG still-picture compression standard," *Communications of the ACM*, vol. 34, pp. 31–46, April 1991.

[13] D. L. Gall, "MPEG: A video comparison standard for multimedia applications," *Communications of the ACM*, vol. 34, pp. 47–58, April 1991.